PoliSave: Efficient Power Management of Campus PCs

PoliSave: Efficient Power Management of Campus PCs

Luca Chiaraviglio, Marco Mellia
Politecnico di Torino
Corso Duca degli Abruzzi 24, Torino, Italy
Email: firstname.lastname@polito.it

***Abstract*—In this paper we study the power consumption of networked devices in a large Campus network, focusing mainly on PC usage. We first define a methodology to monitor host power state, which we then apply to our Campus network. Results show that typically people refrain from turning off their PC during non-working hours so that more than 1500 PCs are always powered on, causing a large energy waste. We then design PoliSave, a simple web-based architecture which allows users to schedule power state of their PCs, avoiding the frustration of wasting long power-down and bootstrap times of today PCs. By exploiting already available technologies like Wake-On-Lan, Hibernation and Web services, PoliSave reduces the average PC uptime from 15.9h to 9.7h during working days, generating an energy saving of 0.6kW/h per PC per day, or a saving of more than 250,000 Euros per year considering our Campus University.**

I. Introduction

Energy consumption has become a key challenge in the last few years. According to [1], the Information and Communication Technology (ICT) sector alone is responsible for a percentage which varies widely between 2% and 10% of the worldwide energy consumption.

Considering the personal computers, manufacturers have focussed their attention to offer energy efficient devices, proposing "green component" as a competitive gain. From a system point of view, even commercial solutions like [2] are becoming to be adopted, and solutions that rely on the idea of protocol proxying [3] and virtualization techniques to concentrate the number of PCs (or functionalities) on to a small set of devices are being investigated. However, the power consumption of a PC, even if used as a "dumb" terminal, is far from being negligible, and today a simple desktop PC requires about 100W to be simply up, despite its much more energy efficient design. Moreover, people generally leave their PC always powered on, even if not used.

In this paper we experimentally investigate the users' habits in our Campus. In particular, we find out that most people prefer to leave their PCs always powered on, causing an energy waste that overall can correspond to more than 250,000 Euros per year. This is mainly due to two dominant factors: i) the little sensibility people have versus the energy cost, and ii) the cost both in terms of time and technical skills to properly and quickly power down and up a PC. These somehow surprising facts suggested us to design a solution that controls the power state of PCs in the Campus, explicitly targeting the ease of use. Even though modern OSes offer tools to remotely control the power state of PCs, we found that these solutions can hardly be applied in our Campus, since: i) we deal with an heterogeneous scenario with different OSes, and ii) users prefer to control their office PC through a simple interface, avoiding the complex OS configuration mechanisms. We therefore designed PoliSave, a centralized web-based architecture which allows users to automatically schedule power state of their PCs. In particular, a server remotely triggers power-up and power-down events by piloting a custom software which has to be installed on each PC. The client software handles all the tasks of correct PC configuration, enabling Wake-On-Lan (WoL) on network cards and hibernation feature on the OS, according to which the current PC state is saved on the hard disk for quick recovery at bootstrap. While the proposed scheme follows a traditional approach, its implementation faced several issues that we describe and discuss in this paper.

To the best of our knowledge, this is the first work which quantifies the energy waste due to PCs left powered on during non-working periods in large Campuses, a timely problem that we target by proposing a solution whose primary goal is to minimize the installation and management problems for users.

All the functionalities of PoliSave have been implemented, and a deployment trial has been studied. Results show that the possible power saving is huge, with negligible impacts on users' habit. At the time of writing, results are so encouraging that PoliSave is being extended to the whole set of Campus PCs, and other Italian Universities are studying how to deploy it. Software is also made available as Open Source from [4].

The paper is organized as follow: Sec. II details the methodology and results to quantify the power consumption of electronic devices in our Campus. Then, Sec. III reports the description of PoliSave architecture. Results are instead presented in Sec. IV. Sec. V describes the related works. Finally, conclusions are drawn in Sec. VI.

II. Monitoring PC usage

To answer the question "How much does the energy consumed by PCs cost in a Campus?", we started collecting data from Politecnico di Torino. Our institution is the second largest technical University in Italy, with about 1800 staff members, and 28.000 students. More than 9000 total PCs are registered

This work has been supported by the WiFi4Energy project funded by the Regione Piemonte.

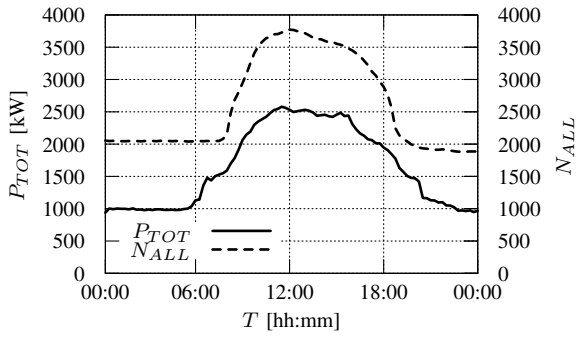

Fig. 1. Total power consumption vs powered on devices during a day.

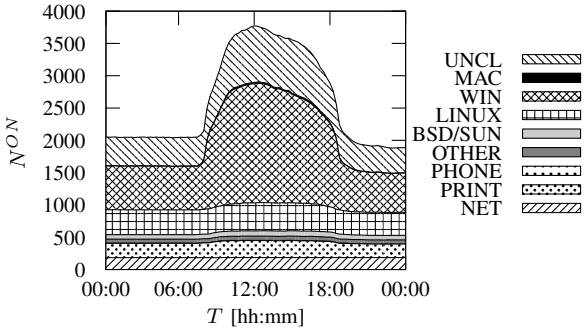

Fig. 2. Variation of powered on devices considering the different OSes.

in our DNS database (DNS-db)[1]. Thanks to available historical data, we tracked the total energy consumption and the electricity bill for our Campus since January 1993. Not surprisingly, the energy consumption has almost doubled, passing from 500 MWh/month in 1993 to more than 1 TWh/month in 2009, with a percentage growth of more than 116%. The electricity price has experienced an even larger increment, with an increase of 218% since 1993, so that now the monthly bill is always more than 150k Euros per month. This is due to the recent increase of the energy cost, which went from 0.009 to the 0.017 Euro per kWh. It is easy to assume that these phenomena are due to proliferation of electronic systems in our Campus, which started to happen since 1996.

These numbers suggest that saving on energy consumption has also beneficial impacts on cost reduction. But "how much of the energy consumption is due to PCs, and how much is *wasted* when PCs are left powered on but idle?" This second question motivated the development of a methodology to monitor PC usage in our Campus.

For our purposes, we used `nmap` version 4.85beta9 [5] to find the number of devices actually powered on. The port-scanning routine is scheduled every $\Delta T = 15m$ using a `cron` entry, to allow `nmap` to complete a scan without time overlap. The port-scanning is performed targeting subnets of each Department of our Campus. In particular, we select a subset of TCP ports to limit the intrusiveness of the scanning while having the highest accuracy. Due to the lack of space we refer the reader to [6] for the complete description. In brief, the probing is performed on 6 ports, limiting the network overhead to 11.52kBps to complete the scan.

The final monitoring of our Campus network has been up and running since mid of June 2009. From the same period, we collected also data regarding the power consumption of all our Campus, and of specific departments as well[2]. Therefore, the comparison between the Campus power consumption and the number of active devices is possible. Fig. 1 reports the average power required by our Campus, P_{TOT}, and the total number of networked devices powered on, N_{ALL}. Results refer to Friday the 26th of June. Both curves follow a typical day-night trend. In particular, the power consumption during the night is 38% of the peak hour demand, i.e., 1MW versus 2.5MW of power. Considering N_{ALL}, during the day more than 3500 devices are powered on. Astonishingly, during the night between Thursday and Friday more than 2000 terminals are left powered on, and on Friday night, no less than 1840 hosts are left up and running for the whole week-end. Update data are available from [4] which tracks the power state of networked devices in real time.

A natural question arises: "How many of the active devices are actually PCs that can be potentially turned off during inactivity intervals?" To answer this question, Fig. 2 reports the breakdown of the active devices, detailing different OS architectures. In particular, for each IP address we perform a double check using both the information registered in the Campus DNS-db and the OS fingerprint feature of `nmap`. IP addresses are then grouped according to different OS categories. From the bottom, the plot reports: network devices (e.g., switches and routers), networked printers, VoIP phones and other small network boxes (e.g., Access Points, small routers, etc.). All these devices are always powered on, with only printers that are seldom powered off at night. Considering Unix like OSes, we define two classes: hosts running Linux and other Unix hosts (mainly BSD/SUN systems). Also in this case, most of Unix hosts are left up and running, possibly due to their "server" capabilities, even if a large fraction of the 350 hosts running Linux could be used as simple terminal. Finally, the largest fraction of devices is due to personal computers running Windows family OSes, representing about 30% of active hosts during the night and more than 40% during the day. Moreover, we notice that this estimation is a lower-bound, since it is very likely that unclassified machines (labeled UNCL) belong to this category as we verified by manually checking a random samples of them[3]. Windows machines are characterized by a more pronounced variability, yet during off-peak periods about 50% of them are left powered on. We argue that a large fraction of Windows hosts that are powered off are actually laptops, while the majority of PCs that are powered on are regular (and more power hungry) desktop PCs. These data confirm that of the 2000 hosts left up and running after-hours, up to 75% could be effectively turned off to save energy.

To assess the impact on the energy consumption of the net-

[1]For security reasons, information about Campus PCs and users are stored in the DNS database as additional fields.

[2]Power consumption data are collected every 15 minutes using probes connected to the main power cabinets. We thank the Electrical Engineering Department for sharing these data.

[3]OS fingerprint is unreliable in case a firewall is present.

TABLE I
POWER CONSUMPTION OF DEVICES

Type	Power [W]
Win	150
Linux	150
UNCL	150
Network	100
Printer	50
BSD/SUN	200
Other	50
Mac	100
Phone	10

worked devices, we estimate the power required by each category of devices according to figures that are publicly found. Tab. I reports the average power consumption estimation we adopted. In particular, we assume that desktop computers consume about 150W to account for the power required by the monitor too. We assume that BSD/SUN systems are used as servers, for which the power footprint is higher than desktop PCs. Finally, the power consumption of network devices like routers is assumed to be 100W, which is possibly a low figure; however routers typically have several interfaces/IP addresses, so that the same device is counted several times during a scan process. Fig. 3 reports the power consumption projection, considering the dataset of Fig. 1. During the day the total energy required is more than 500kWh, representing around 26% of the total power required by our Campus. During the night, about 300kWh are still consumed, corresponding to 35-40% of the total power consumption (see Fig.1). Moreover, the power consumption of PCs (both Windows and Linux) is predominant, suggesting that further improvements have to be considered to reduce their power consumption.

Finally, we conducted a survey among users to investigate what are the reasons that refrain users from turning off their PC. First, the economic incentive is little or totally absent in the context of a Campus, since energy costs are not split among users. Second, the frustration of long power down and bootstrap times of today PCs typically discourages the adoption of energy wise policies by users. Third, the loss of state a reboot causes has also been found to be annoying (if not upsetting), since users prefer to leave the office with applications and documents still opened on their PCs' desktop. Fourth, administration tasks (e.g., software updates or deployment of new software) can be scheduled at night. And fifth, some users want to access the applications and data on their office PCs even when they are at home. While technical solutions to the previous issues are already available (e.g., the "hibernate" feature that allows to freeze and recover the state of the PC storing its state on a file), people are not aware of them. In addition, the OS configuration has been identified as a complex task that typical user prefers to avoid. Note that some of these motivations were already pointed out [7], in which authors consider a set of home users.

III. POLISAVE: MANAGING PC ENERGY CONSUMPTION

The design of PoliSave has to face a complex and very heterogeneous scenario. In particular, we address the following requirements: 1) **heterogeneity**, both in term of users and OSes; 2) **remote control**, since power management actions need to be performed remotely; 3) **simple GUI**, since the complexity of the control panel offered by OSes was identified as one of the major problem; 4) **custom deployment**, since the software can be either manually or automatically installed; 5) **security** has to be guaranteed since the actions are performed remotely; 6) **consistent information**, since the software has to handle the association between a PC and its user.

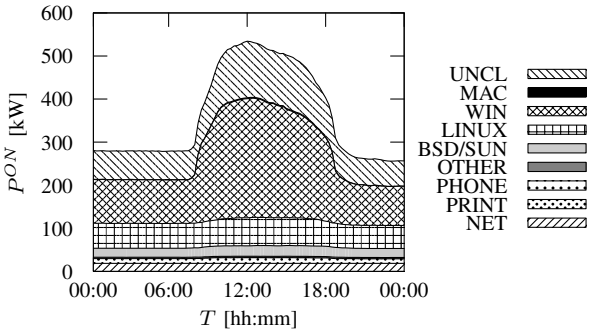

Fig. 3. Estimation of the total power consumed by devices.

These requirements call for a new solution, since the available ones fail to address some of constraints. For example, a possible solution for Windows family OSes might be to create a domain policy in the Active Directory services. However, this technique can not be applied in our Campus, since not all Windows PCs are registered in a domain.

Given the previous presented list of requirements, we then sketch the final *PoliSave architecture*. It is based on three main components: a Server, a Client, and a Communication Protocol. Due to the lack of space we refer the reader to [6] for a detailed description. Additionally, the live monitoring tool described in Sec. II has been integrated among the services of our Campus.

1) Client Architecture: The client manages the actual powering off mechanism. Two different client architectures have been developed, for Windows and Linux systems respectively.

Considering the Windows version, the client has been implemented as a multi-threaded background service. One thread manages the server communication, while a second thread is used to display pop-up messages to communicate with the user. Pop-ups are used to warn the user when an action is going to be performed, allowing him to override it. To perform the PC shutdown, the client exploits the standard Windows (or Linux) API, which requires the program to be executed as administrator. Therefore, we force the execution of the client with high privileges during the installation phase.

The Linux version of PoliSave is composed by a background daemon, named `polisaved`. `polisaved` communicates with the server through socket function calls and displays the informations to the user by means of pop-up windows. The pop-up is activated using `dbus` and the `x11` system, eventually opening multiple pop-ups in case several users are remotely connected. Actions are instead performed by invoking the primitives of the Hardware Abstraction Layer (`HAL`), which allows to list hardware properties, and control power state of the PC.

```
bool ClientSetup(bool silent, INFO info) {
    RetrieveInfo(&info);
    SendInfo("START",info,&s_ans,&ip,&mac);
    if((s_ans=="NO") && (!silent)) {
           ShowHelp(info);
           return false;
    }
    ADAPTER adapt=FindAdapt(ip,mac);
    if(info.vectAdapt[adapt].wol==false)
       SetWol(adapt);
    if(info.Hib==false)
       SetHib(info);
    InstallPolisave(info);
    RetrieveInfo(&info);
    SendInfo("END",info,&s_ans);
    if((s_ans=="NO") && (!silent)) {
           ShowHelp(info);
           return false;
    }
    ShowWebPage(info);
    return true;
}
```

Fig. 4. The installation steps of the PoliSave client.

During setup, the client software performs some preliminary actions to optimally configure user's PC. Fig. 4 reports the main steps. In particular, the installer gathers system information, including WoL state, Hibernation capabilities, IP and MAC addresses of all network interfaces, OS type and version, and DNS name. This information is sent to the server, using either HTTP or HTTPS protocol. Then, once the server answer is received, the client processes it: if the server does not grant the client, the setup is aborted and a web page with detailed information is presented to the user. In case the server grants the installation, the IP and MAC address of the selected network interface (the one actually used to contact the server) are given back. The installer then enables the WoL for this interface and activates the OS hibernation feature; finally, the PoliSave service (or daemon) are installed. Result of each operation is sent back to the server, which finally records the successful client installation, and presents the user its intranet private area to show the installation result. For example, in case the WoL or the Hibernation have not been activated, detailed instructions are displayed to help the user. The client can be also installed in silent mode: in this case an eventual failure is reported directly to the PC administrator and no information is displayed to the user.

At the moment of writing this paper, the client supports Windows OS (Windows 7, Vista and XP, 32 or 64 bits, and earlier versions like Windows 2000 and Windows 98). Considering Linux, a complete version for the Ubuntu distribution has been tested.

2) Server Architecture: The server is the core of PoliSave. It performs client remote power control and it manages the database of clients, which includes the scheduled events of users. Both powering off and powering on operations are managed by the server to which users can access using a web interface. Users are free to specify a timetable that stores scheduled *actions* like *stand-by, hibernation, power-off, power-on* of a PC. The server then automatically performs the operations, periodically querying the database to look for

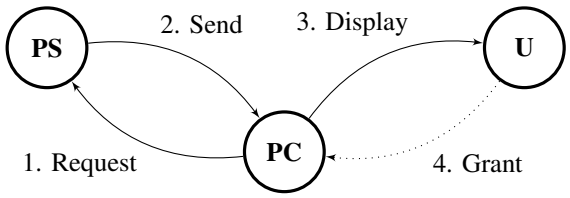

Fig. 5. The powering off protocol of PoliSave.

actions to be performed. In addition, the user can perform real time actions, e.g. to immediately turn on (or off) the PC, triggering a "manual action".

The web interface allows users to interact with the server, from which users can change/add/remove entries from the PC power scheduling. Web pages are integrated in the Intranet personal pages, so that the user can simply login using his credentials. The web interface shows the list of PCs the user has administrative or user credentials, showing the information contained in the Campus DNS-db, i.e., IP and MAC addresses, administrator's and PC name. The web site is public to remotely control PCs even from outside the University Intranet.

Maintaining the Campus DNS-db updated is a major issue, since people typically forget to update the information when changing hardware or setup. To help updating it, during installation the server verifies the client IP and MAC addresses in the DNS-db. If an entry is returned and the data are consistent, the installation proceeds. If instead there is some mismatch, an email is sent to warn both user and administrator, and installation of the client program is aborted. Once the client software has been successfully installed, the DNS-db is verified and eventually updated every time a client contact the server, thus allowing to track an eventual modification of the client IP or MAC addresses.

The server is implemented using simple Visual Basic scripts, with a SQL Server back-end to store the scheduling and additional information. As in any server based architecture, the server represents a single point of failure, and standard solutions are available to eventually handle server failures.

3) Communication Protocols: The *power-on* mechanism relies on the WoL protocol standard. According to the standard, the PC is shut down (Sleeping, Hibernating or Soft Off, i.e. ACPI state G1 or G2), with power reserved for the network card. The network card keeps listening for a specific packet containing a message with 6 copies of its MAC address, called the "Magic Packet," broadcasted on the LAN. The magic packet has to be routed over the network of our Campus which is a L3 network. However, for security reason, routers typically do not forward broadcast messages, so that proper configuration is required on all L3 routers, since the broadcast packet must be forwarded to the destination subnet too.

The *power-off* technique is instead implemented by a proprietary protocol, as Fig. 5 shows. More in depth, the client (PC) performs a *periodic polling* with the server (PS), by sending a *Request* message, whose fields are detailed in Tab. II. The information sent to the server includes the list of IP and MAC addresses for all the network interfaces of the host, the PC

TABLE II
STRUCTURE OF THE MESSAGES

Request	IP list, MAC list, Counter, Host, OS info
Send	Action, Time, *URL*, *Message*

name as it appears in the DNS and the type of OS in use, including the system version. These fields are used to keep the Campus DNS-db updated. Finally, a counter, increased at any polling and reset after any powering on operation, is reported too. This counter allows to track the PC power state, and eventually detect missing poll operations.

When PS receives a *Request* message, it sends back a response containing an *action* among {*Hibernation, Stand-by, Power-Off, Wait, Message*}. Besides the power control actions, the $Wait$ action forces the client to simply return idle and wait for the next polling event. The $Message$ action allows the server to send a string *Message* that it is displayed on the user screen. This feature can be used for example to warn users to update the client. The optional URL field contains the new server URL that has to be used for future polling (useful to relocate the server). In all cases, the response message includes a $Time$ value which details when the client has to perform the next poll. This parameter trades between timely management and server load. In particular, low $Time$ values lead to quick response times, but tend to increase the load on the server.

When an action has to be performed, the client displays a warning message to the user (U) via a pop-up window. The user can optionally cancel or grant the action, up to a maximum timeout of one minute. If the event is granted (or the timeout has expired), then the action is performed by invoking the system APIs and finally the PC is shut down.

All protocol messages are encapsulated using HTTP protocol, and `OpenSSL` libraries are included to guarantee privacy and authenticate the server. In this case the HTTPS protocol is used, and the public key of the server is distributed with the setup package, together with the server's certificate.

Thanks to the fact that the communication is started by the client, the powering off mechanism works even if the client is behind a NAT. Another advantage is that the client OS does not need to keep any port open, avoiding both security issues, and firewall configurations. The powering on procedure still relies on WoL messages sent by the server, so that the eventual NAT box has to appropriately route them. Finally, a possible drawback of the communication protocol is that a manually triggered powering off event will be actually delayed until the next client polling phase starts.

IV. RESULTS

To test the effectiveness of PoliSave, we installed the system on a trial of 70 users of the Electrical Engineering Department (EEDept). The aim was (i) to test the software implementation on a real environment, (ii) to assess the possible power-saving and (iii) to collect feedback from actual users. In particular, we start by providing a detailed characterization of each device power state, i.e., the uptime of a device. We assume that time is divided in slots of duration $\Delta T = 15m$, corresponding to the measurement instants, i.e. $i\Delta T$ is the current slot, $i \in \{1, 2, \ldots, 96\}$ to consider single day. For each device x, we define the function

$$I_x(i) = \begin{cases} 0 & \text{if } x \text{ is OFF} \\ 1 & \text{if } x \text{ is ON} \end{cases} \quad (1)$$

Then, we compute the total amount of times x is on in a given day $\mathcal{D}$:

$$ON_x(\mathcal{D}) = 15m \sum_{i \in \mathcal{D}} I_x(i) \quad (2)$$

Finally, we average over the set $\mathcal{K}$ of days in the dataset, obtaining the average time the device x is active per day, i.e., the host daily average uptime:

$$ON_x = \frac{\sum_{\mathcal{D} \in \mathcal{K}} ON_x(\mathcal{D})}{|\mathcal{K}|} \quad (3)$$

We have computed both $ON_x(\mathcal{K})$ for $x \in \{PoliSave\}$ and $ON_x(\mathcal{K})$ for $x \notin \{PoliSave\} \cap \{EEDept\}$; $\mathcal{K} = \{Mon, Tue, Wed, Thu, Fri\}$.

Fig. 6 (left) shows the comparison among the two CDFs. Normally, about 53% of PCs without PoliSave are always on, while with PoliSave this percentage falls to less than 6%, with most of PCs that are alive for less than 12h. The average daily uptime of PCs managed by PoliSave is $9.7h$, while the average daily uptime for other PCs is $15.9h$. This corresponds to an average saving of more than $6h$ per working day, or an annual saving of about 219kW/year (about 100,000 Euros per year using current electricity costs). The savings achieved including weekends, for which PoliSave PCs result powered off for the whole day with probability 0.93, amount to more than 250,000 Euros per year.

To give some more details on the users' habit, Fig. 6 (center) reports the number of ON and OFF events recorded during one week, computed as

$$E^{OFF}(i) = \sum_{x \in \mathcal{X}} |(I_x(i) == 1) \;\&\&\; (I_x(i-1) == 0)|$$
$$E^{ON}(i) = \sum_{x \in \mathcal{X}} |(I_x(i) == 0) \;\&\&\; (I_x(i-1) == 1)|$$

with $\mathcal{X} = \{PoliSave\}$. Most of PCs running PoliSave are powered on in the morning and turned off during the late evening, with the variability of the measurements that suggests each user has customized the action from the web-interface. Interestingly, some rare events are recorded at night, suggesting that the option of manually turning on/off the PC via the server web interface is seldom used.

Finally, for each PC running PoliSave we compute the probability to be alive during a time slot i as

$$Prob\{ON_x(i)\} = \frac{\sum_{k \in \mathcal{K}} I_x(i + k24)}{|\mathcal{K}|} \quad (4)$$

Examples of $Prob\{ON_x(i)\}$ are plotted in Fig. 6 (right), which shows three different profiles. In particular PC_2 exploits automatic power-on feature of PoliSave during all working mornings at 8.30am. The manual feature is instead adopted for powering off the PC. The probability of finding PC_2 on during the day is then $5/7 = 0.71$. On the contrary, the profile

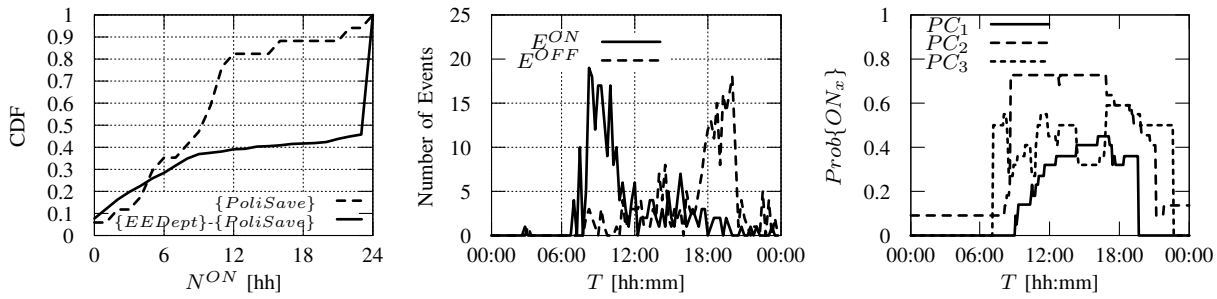

Fig. 6. (left) Host daily average uptime for PCs in the trial and for other PCs. (center) Number of on and off events recorded every 15 minutes. (right) Probability to be on for three PCs using PoliSave.

of PC_1 reveals that PoliSave automatically turns off PC_1 at 19:30, so that the probability to be on at night is equal to 0. Manual powering on is adopted, with the user being in the office (its PC being alive) smaller than 0.5. PC_3 leverages instead both the powering on and off features, and a very aggressive policy is adopted by the user to turning off the PC when possibly idle.

These very favorable results actually encourage our effort, and, at the time of writing, PoliSave is being extended to the whole set of Campus PCs, and other Italian Universities are studying how to deploy it.

V. Related Work

In [8] the authors collected data on the after-hours power state of networked devices in office buildings, showing that most of devices are left powered on during night. However, the proposed measurement technique is manual, thus limiting the number of measurements over time and the applicability in large buildings. In this paper instead we have applied a fully automatic technique that scales well also for large networks and tracks the number of devices powered on in real time.

A complementary approach to put the device in power save mode relies on the idea of proxying. In [9] the authors propose the proxying technique for network elements. In [10], [11] the authors extend their technique to end-user PCs, analyzing which protocols and applications require proxying. Moreover, connectivity issues are considered in [3]. All of these works show that the possible power saving derived from proxying can be huge. However, this technique requires the modification of the hardware on PCs, which can be an hard task in large Campuses. Our solution instead is completely software-based to explicitly address the ease of management.

Finally, commercial vendors like [2] are proposing green solutions to reduce the power consumption waste for enterprise networks. Cisco Systems and IBM have recently announced a solution to remotely monitor and control network devices such as Access Points, routers, phones and even terminals, exploiting the idea of a central administration [12]. Unfortunately, at the time of writing, this solution supports only Cisco network devices, and server OSes, while ignoring Desktop PCs.

VI. Conclusions and Future Work

We have presented PoliSave, a software designed to reduce power consumption in computer networks. We have first quantified the energy waste that idle PCs generate in our Campus, showing that more than 50% of PCs are always powered on. We then described and discussed the architecture of PoliSave, a web-based service that allows users to automatically schedule powering on and off of their PCs. Finally, we have performed extensive measurements on a trial involving more than 70 users, proving the effectiveness of our solution. PoliSave is being extended to all PC in our Campus, with the goal of saving about 250,000 Euros from the University energy bill.

As future work, we want to customize the monitor capability of PoliSave, so that individual users can track the power consumption of their PCs. Another future topic is to introduce active learning techniques in order to track the user activity and then automatically compute the best power scheme to apply for each user.

References

[1] SMART 2020 Report, *Enabling the low carbon economy in the information age*, http://www.theclimategroup.org.

[2] Verdiem Surveyor, http://www.verdiem.com.

[3] M. Jimeno, K. Christensen, and B. Nordman, "A Network Connection Proxy to Enable Hosts to Sleep and Save Energy," *IEEE IPCCC '08*, Phoenix, AZ, December 2008.

[4] PoliSave web site, http://www.polisave.polito.it/hosts_new.shtml.

[5] `nmap` Security Scanner, http://nmap.org

[6] L. Chiaraviglio, M. Mellia, "PoliSave: Efficient Power Management of Campus PCs," *Technical Report No.01012010*, Politecnico di Torino, Italy, January 2010. Available from http://www.telematica.polito.it/chiaraviglio/papers/polisave_techreport.pdf.

[7] M. Chetty, A.J. B. Brush, B. R. Meyers, P. Johns, "It's not easy being green: understanding home computer power management," *ACM CHI '09*, Boston, MA, April 2009.

[8] J. Roberson, C. Webber, M. McWhinney, R. Brown, M. Pinckard, J. Busch, "After-hours Power Status of Office Equipment and Energy use of Miscellaneous Plug-load Equipment," *Report LBNL-53729-Revised*, LBNL, Berkeley, CA, 2004.

[9] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation," *USENIX NSDI '08*, San Francisco, CA, 2008.

[10] S. Nedevschi, J. Chandrashekar, B. Nordman, S. Ratnasamy, N. Taft, "Skilled in the art of being idle: reducing energy waste in networked systems," *USENIX NSDI '09*, Boston, MA, 2009.

[11] Y. Agarwal, S. Hodges, J. Scott, R. Chandra, P. Bahl, R. Gupta, "Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage," *USENIX NSDI '09*, Boston, MA, April 2009.

[12] Cisco Energy Wise, http://www.cisco.com/en/US/products/ps10195/index.html.